RxJS Operator Handbook — Full Edition

Interactive demos, diagrams, best practices, and pitfalls.

Table of Contents

audit / auditTime

- **Pitfall:** Emits only the *last* value after the silence window. If you expected the first, you likely want `throttle`/`throttleTime`.
- **Best practice:** Use for *sampling on trailing edge* (e.g., mousemove → update preview after user pauses).

buffer / bufferTime / bufferCount / bufferToggle / bufferWhen

- **Pitfall:** Unbounded buffers can grow large.
- **Best practice:** Always choose count/time bounds; drain quickly downstream.

catchError

- **Pitfall:** Swallowing errors hides real failures.
- **Best practice:** Log/telemetry the error; provide *user-meaningful* fallback; consider `retryWhen` with backoff for transient issues.

combineLatest / combineLatestAll

- **Pitfall:** `combineLatest` waits for *each* source to emit at least once; initial UI can appear “stuck”.
- **Best practice:** Pre-seed with `startWith` for initial values.

concat / concatAll / concatMap

- **Pitfall:** Serializing work increases latency under bursty input.
- **Best practice:** Use when **order** matters and backpressure is acceptable; otherwise consider concurrency (`mergeMap` with bound).

debounce / debounceTime

- **Pitfall:** Cancels intermediate values; async tasks inside `switchMap` + `debounceTime` may never fire during steady typing.
- **Best practice:** Great for *typeahead* requests; choose a delay that balances latency and load.

defer

- **Pitfall:** Factory re-runs on each subscription—be mindful of side effects.
- **Best practice:** Wrap resource creation (e.g., `fetch`) to ensure fresh state per subscriber.

delay / delayWhen

- **Pitfall:** With hot sources, `delay` shifts values but won’t pause the source; you can still overwhelm consumers.
- **Best practice:** Use for UI polish or backoff visuals; for rate limiting, prefer `throttle`/`audit` patterns.

distinct / distinctUntilChanged / distinctUntilKeyChanged

- **Pitfall:** `distinct` tracks *all* seen keys → memory growth.
- **Best practice:** Prefer `distinctUntilChanged` for streams with many repeats; pass custom comparer carefully.

exhaustMap / exhaustAll

- **Pitfall:** Ignores new emissions; users can feel “button doesn’t work.”
- **Best practice:** Ideal for *form submit* or *login* to prevent double-submission.

expand

- **Pitfall:** Can generate infinite trees; ensure termination or `take`.
- **Best practice:** Breadth-first recursion for pagination/tree walks with guards.

forkJoin

- **Pitfall:** Waits for all sources to complete; **never emits** if a source doesn’t complete.
- **Best practice:** Use for one-time “join at the end”; for live combining, use `combineLatest`.

groupBy

- **Pitfall:** Creates many inner subjects; leaks if not consumed.
- **Best practice:** Flatten with `mergeMap`/`concatMap` and `finalize` per group.

merge / mergeAll / mergeMap

- **Pitfall:** Unbounded concurrency = resource spikes and race conditions.
- **Best practice:** Bound concurrency (`mergeMap(fn, n)`); preserve order with `concatMap` when needed.

pairwise

- **Pitfall:** First emission is skipped; don’t expect pair on the first value.
- **Best practice:** Use where deltas matter (scroll, cursor movement).

publish / multicast / publishReplay / publishBehavior / publishLast

- **Pitfall:** Legacy; manual `connect()` and lifecycles are error-prone.
- **Best practice:** Prefer `share` / `shareReplay`; if using, ensure `connect()` and teardown paths are clear.

race

- **Pitfall:** Fastest to *first emission* wins, others are cancelled—might not match “first complete” semantics you expect.
- **Best practice:** Use for *first-response wins* UX.

scan / reduce / count

- **Pitfall:** `reduce` emits only on complete; don’t expect intermediate values.
- **Best practice:** Use `scan` for live state, `reduce` for final aggregates.

share / shareReplay

- **Pitfall:** `shareReplay({refCount:false})` can keep subscriptions alive forever → memory leaks.
- **Best practice:** In UI, prefer `shareReplay({ refCount: true, bufferSize: 1 })` when caching latest; understand hot vs cold sources.

startWith / withLatestFrom

- **Pitfall (`withLatestFrom`):** No output until the “other” has emitted at least once.
- **Best practice:** Pair with `startWith` to seed initial value.

switchMap / switchAll / switchScan

- **Pitfall:** Cancels previous inner; **loses** response if late (race with network).
- **Best practice:** Great for typeahead; for “every request must finish,” use `concatMap` or `mergeMap` with limits.

take*, skip*, elementAt

- **Pitfall:** `takeUntil` requires notifier to fire; leaks if notifier never triggers.
- **Best practice:** Couple with component lifecycle notifier (e.g., `destroy$`).

throttle / throttleTime

- **Pitfall:** Default drops trailing values.

- **Best practice:** Use `{ trailing: true }` if last value matters; otherwise `auditTime` for trailing edge.

timeout / timeoutWith

- **Pitfall:** Timing out stateful processes may leave partial state.
- **Best practice:** Combine with `retryWhen` backoff; ensure operations are idempotent.

window*

- **Pitfall:** Windows not drained → orphaned subscriptions.
- **Best practice:** Immediately flatten windows with `mergeAll` or `switchMap`.